최종 보고서

12161618 컴퓨터공학과

육상은

1. 개발 계획

   A. 개발 환경: UI구현을 위해 PyQT5 모듈 사용하여 개발툴 pycharm에서 python3.6.8 로 작성

2. 설계 세부 사항

   A. 오목게임 분석

      ① 오목: 흑과 백의 2인이 오목판 가로, 세로 15*15의 선으로부터 225개의 교차점에 교대로 한 수씩 두어 먼저 연속적으로 돌 5개를 만들면 승리하는 게임

      ② 전형적인 two-play game이다.

      ③ 일반적으로 흑이 먼저 시작한다. (흑이 유리하므로 정중앙에 배치하고 시작)

      ④ 33금지 유무를 경기시작전 선택할 수 있다.

      ⑤ 삼: 스트레이트 사를 만들 수 있는 상태

      ⑥ 스트레이트 사: 돌이 연속적으로 4개 존재하며 오목을 만들 수 있는 상태

      ⑦ 무승부 성립 유형: 흑과 백이 연속적으로 착수를 패스하였을 경우 or 오목판에 더 이상 이길 수 있는 경우가 존재하지 않을 경우 or 오목판이 모두 꽉 찼을 경우

   B. 개발 방법 및 알고리즘 설명

      - 유저 인터페이스 (pyQt5 모듈을 이용)

      ① 흑(선수) 백(후수) 선택 기능 구현

      ② 난이도 3단계(초급, 중급, 고급) 조절 메뉴 구현

      ③ 바둑판 및 바둑알 시각화 및 마우스 클릭으로 바둑알을 배치시키는 기능 구현

      ④ 33 반칙 on/off 기능 구현 (실제 적용은 불가하고 선택만 가능)

      ⑤ 알고리즘 선택 기능 구현

      ⑥ replay할 수 있도록 다시 시작 버튼 구현

      ⑦ 인공지능이 3초안에 돌을 배치시키도록 구현

      - 난이도 조정방법

① Depth로 조정 – 초(1), 중(2), 상(3)으로 설정

- 33금지 (on일 경우) (구현안됨)

① 알고리즘의 경우 탐색 중 33금지 경우 제외하고 탐색

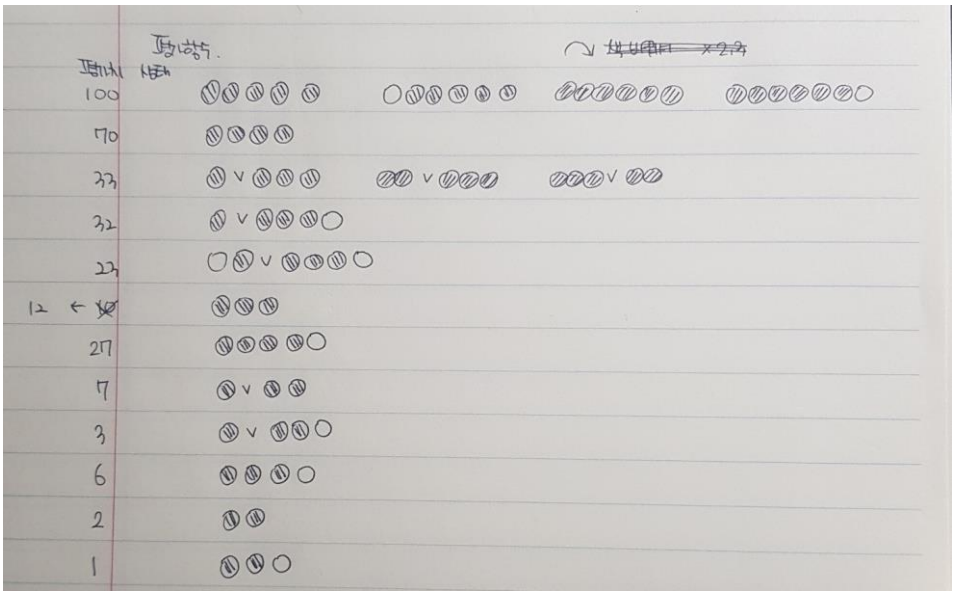
② 사용자가 금지된 자리에 바둑알 배치시킬 경우 경고 창 띄우기

- 평가함수

① 상대방의 바둑알 위치 및 상태에 따라 평가치를 다르게 하여 설정

② 상대방의 바둑알 상태가 오목에 가까워질수록 평가치를 높게 설정

③ 검은색 바둑알의 state에 따른 평가함수 표



④ 흰색 바둑알의 state에 따른 평가함수표는 위에 표에서 검은색, 흰색을 바꿔서 보면 된다.

⑤ 만약 컴퓨터가 선수(흑)일 경우 검은알 state의 값은 양수로 유지하고 흰색알 state의 값은 음수로 바꾼 후 곱하기 2.4를 해서 적용시킨다. (흰색알에 대해 방어를 더 잘하도록 하기 위해)

⑥ 만약 컴퓨터가 후수(백)일 경우 5번을 반대로 하면 된다. 검은알 값을 음수 * 2.4, 흰색알 값을 양수로 해서 오목판 상태에 따른 총 평가치를 계산하다.

- 알고리즘 구현

① MiniMax 알고리즘

Max node: 인공지능이 바둑알을 배치시켰을 때 이길 가능성 큰 값

Min node: 인공지능의 상대편이 바둑알을 배치시켰을 때 인공지능이 이길 가능성 가

장 작은 값

Root 노드는 현재 board의 상태를 나타내며 max node이다. 이후 다음 깊이 노드들에 현 board상태에서 인공지능이 바둑알을 배치할 수 있는 모든 경우에 대해 노드들을 생성하고 이 노드들은 min node들이다. 다음 깊이에서는 min node들에 대해 인공지능이 바둑알 배치한 상황에서 사람이 바둑알 배치할 수 있는 모든 경우에 대한 node들을 생성한다. 이를 반복적으로 수행하여 말단 node까지 이동 후 말단 노드에서 평가치를 계산한다. (bfs 알고리즘을 이용)

② MiniMax with alpha-beta pruning 알고리즘

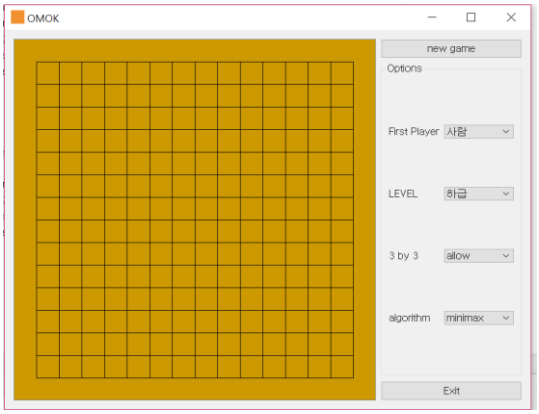
위에서 설명한 minimax 알고리즘과 같은 방법으로 수행되며 minimax알고리즘 구현 시 탐색 공간 및 범위가 광범위해 탐색하는데 걸리는 시간이 길다. 이를 해결하기 위한 알고리즘으로 각 노드마다 [alpha, beta] 범위를 설정한 뒤 alpha >= beta가 되는 경우 즉 이후 검사할 필요가 없는 상황에서 pruning을 하는 알고리즘이다.

탐색할 때 걸리는 시간을 줄여준다.

3. 프로그램 및 코드 설명

A. 프로그램 작동 방법

- Pycharm개발툴을 실행시키고 실행python 코드에 대해 shift+10 or Run 버튼을 누른다.



- 위와 같이 오목 GUI 화면이 생성되고 오른쪽 option메뉴에서 first player, LEVEL, 3 by 3, algorithm을 선택한다. 메뉴를 새로 선택할 때마다 board가 초기화된다.
- New game버튼을 누르면 게임을 다시 시작할 수 있다.
- 프로그램을 종료하고 싶은 경우 오른쪽 하단에 exit 버튼을 누르면 된다.
- 게임은 board에 원하는 곳에 바둑을 배치하면서 시작된다.

B. 코드 설명

- 전역 변수(고정된 값들을 전역변수로 설정한다): 화면 너비(width, height), board길이(board_l), 흑돌(BLACK), 백돌(WHITE), LEVEL(HIGHT, MID, LOW), minimax알고리즘(MINIMAX), alpha-beta(ALPHABETA)
- BLACK_STATE: 흑돌의 상태에 따른 state와 score가 입력된 배열. 평가치에서 사용한다.
- WHITE_STATE: 백돌의 상태에 따른 state와 score가 입력된 배열. 평가치에서 사용한다.
- class 내부 변수(ui에 대한 변수): 15*15의 board(grid), level box값(level), 3 by 3 rule(rule), 선수와 후수 표시(human, computer), 사용할 알고리즘(algorithm)
- initUI(self): UI 초기 구현으로 layout과 창에 대해 구현한다.
- Center(self): initUI()에서 사용되는 함수로 창을 화면가운데로 맞춰준다.
- Init_board(self): 오목판 초기화 및 boundary, grid변수를 초기화 한다. 처음 함수 생성시 호출되며 new game button or option을 바꿀 때마다 호출된다

```
#옵션의 combo box들에 대한 정보 처리
#param : combo box의 click된 text
def onActivated(self, text):...
```

```
#화면상 x,y 좌표를 grid위 i, j로 바꾸기
#return : grid위 (i, j)
def xyTogrid(self, x, y):...
```

```
# 바둑돌 배치
# param : grid위 (i, j), 돌의 색상(0이면 흰색, 1이면 검은색)
def draw_baduk(self, x_g, y_g, baduk):...
```

```
# 보드내 클릭된 위치에 HUMAN 오목 배치 후 알고리즘을 통해 AI 오목 배치하는 함수
# param: 마우스버튼 클릭시 좌표
def mousePressEvent(self, e):...
```

```
#주어진 x, y를 통해 경계 다시 설정 - 탐색시 사용될 경계
#param : 새로 등록된 grid 좌표 (x, y) , 기존 경계 (x1, y1, x2, y2)
#return : 새로운 경계 (x1, y1, x2, y2)
def boundary(self, x, y, x1, y1, x2, y2):...
```

```
#alpha-beta pruning algorithm
#처음 시작시 a = [-1, -1, -infinity] , b = [-1, -1, infinity]
#param : 현재 grid상태, 노드 깊이, human or computer, alpha, beta, boundary
#return : min or max 깊이에서의 각 node별 min or max 값과 해당 좌표
def alpha_beta(self, state, depth, player, a, b, x1, y1, x2, y2):...
```

```
#minimax algorithm
# param : 현재 board, node index, human or computer, 탐색 바운더리 (사각형[[x1, y1], [x2, y2]])
# return : 현 state에서 평가함수 값 중 가장 큰 or 작은 값과 해당 좌표 (좌표(x, y), socre)
def minimax(self, state, depth, player, x, y, x1, y1, x2, y2):...

#evaluate function - 평가치를 이용하여 score 계산
#param : 현 grid, boundary
#return : 계산된 평가치
def evaluate(self, state, x1, y1, x2, y2):...

# 현재 board의 상태
# param : 현 grid
# return : 인간승(0), 인간패(1), 비김 or 결정 안됨 (-1)
def outcome(self, state):...

# player의 오목알이 연속으로 5개 있는 것 찾기 - winner
# param : 현 grid, computer or human
# return : player의 승(1), 이외에(0)
def wins(self, state, player):...

if __name__ == '__main__':  # 현재 모듈의 이름이 저장되는 내장 변수
    app = QApplication(sys.argv)  # 모든 pyqt5 어플리케이션은 application 객체를 생성
    ex = MyApp()
    sys.exit(app.exec_())
```

위 함수에서 프로그램이 처음 시행된다.

- 실행 순서: MyApp class 내부에서 순서

I. main함수에서 class 생성시 option조작 변수들 및 board상태에 대해 초기화 후 initUI()함수 실행

II. initUI()함수에서 GUI를 구현 및 실행창을 화면 중심에 설정(center()함수), 오목판을 초기화 해준다(init_board()함수 호출)

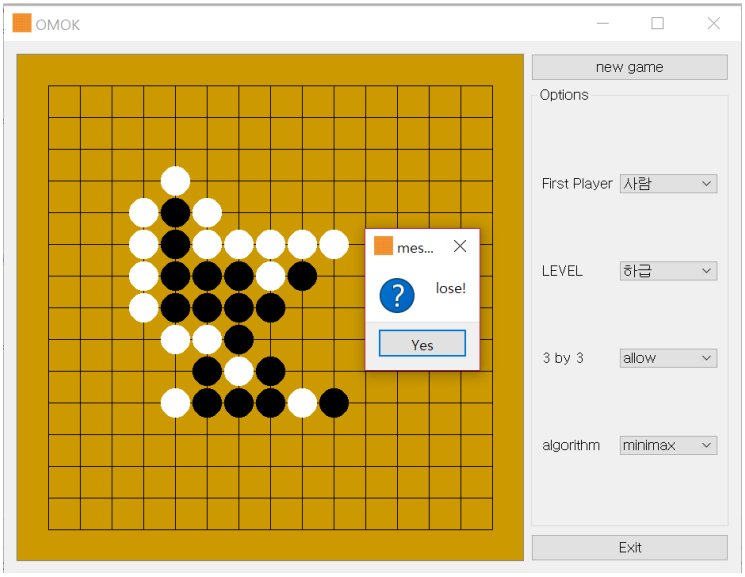
III. new game button 클릭시 init_board()함수로 연결, exit button 클릭시 프로그램 종료, option combo box 클릭시 onActivated()함수로 연결되어 해당 변수 설정 다시 설정하고 함수 내부에서 init_board()를 호출하여 화면을 reset한다

IV. board 클릭시 mousePressEvent()함수로 이동하며 좌표 설정하기 위한 xyTogrid()함수 호출 후 board에 표시하기 위해 draw_baduk()함수 호출, 승패확인을 위한 outcome() 함수 호출, boundary 재설을 위한 boundary()함수 호출, 이후 컴퓨터 차례를 수행하기 위해 minimax() or alpha_beta()함수를 호출한다.

V. 각 알고리즘 함수에서는 재귀함수가 수행되며 평가치를 계산하기 위해 evaluate()함

수가 호출된다.

C. 실행화면 스크린샷



4. 고찰

A. Minimax알고리즘 구현 시 level에 따라 탐색 깊이를 조정할 때 깊이가 1일 경우는 탐색이 빠르지만 깊이가 2 이상일 경우 너무 오래 걸린다.

B. Minimax with alpha-beta pruning 구현 시 minimax알고리즘에 비해 더 나아진 점을 찾지 못한다. (시간적인 면에서), 이론적으로 코드를 잘 구현한 것 같지만 나은점이 없다는 점에서 pruning이 제대로 안되는 것 같다.

C. 평가함수의 경우 중요하게 생각되는 경우들을 배열로 만들어서 다른 값을 적용시킴. 컴퓨터와 모의 대결할 때마다 가능한 경우가 보이면 추가했다.

D. 평가함수의 값에 따라 인공지능이 잘하고 여부가 결정됨. 평가치 조절이 조금 더 필요해 보인다.

E. 시간상 3 by 3을 구현하지 못함. button이나 변수는 생성

F. 3초 이내에 대결이 진행되도록 구현하였으나 3초로 제한할 경우 인공지능이 오목을 제대로 못함 – 따라서 대결을 할 때 난이도를 항상 1로 설정하고 한다. (초급)

G. 사람이 바둑을 배치 후 인공지능이 자신의 바둑을 고려하는 동안 사람의 바둑이 ui화면에 표시가 되야 하는데 인공지능이 자신의 바둑을 고른 후 동시에 사람바둑과 인공지능 바둑이 화면에 나타난다.

5. 참고 자료 – UI구현, minimax 알고리즘, alpha-beta pruning 알고리즘

https://codetorial.net/entry/reimplement2

https://oceancoding.blogspot.com/2019/03/blog-post.html
https://github.com/Cledersonbc/tic-tac-toe-minimax/blob/master/py_version/minimax.py
https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/